Abstract

Online advertising allows advertisers to only bid and pay for measurable user responses, such as clicks on ads. As a consequence, click prediction systems are central to most online advertising systems. With over 750 million daily active users and over 1 million active advertisers, predicting clicks on Facebook ads is a challenging machine learning task. In this paper we introduce a model which combines decision trees with logistic regression, outperforming either of these methods on its own by over 3%, an improvement with significant impact to the overall system performance. We then explore how a number of fundamental parameters impact the final prediction performance of our system. Not surprisingly, the most important thing is to have the right features: those capturing historical information about the user or ad dominate other types of features. Once we have the right features and the right model (decisions trees plus logistic regression), other factors play small roles (though even small improvements are important at scale). Picking the optimal handling for data freshness, learning rate schema and data sampling improve the model slightly, though much less than adding a high-value feature, or picking the right model to begin with.

1. 介绍

1.1 论文背景

本文主要介绍Facebook提出的CTR预估模型LR(Logistic Regression)+GBDT。当时深度学习还没有应用到计算广告领域，Facebook提出利用GBDT的叶节点编号作为非线性特征的表示，或者说是组合特征的一种方式。

LR+GBDT相比于单纯的LR或者GBDT带来了较大的性能提升，论文中给出数据为3%，这在CTR预估领域确实非常不错。除此之外，Facebook还在在线学习、Data freshness、学习速率、树模型参数、特征重要度等方面进行了探索。

相比于搜索广告领域，根据用户query来给出候选广告，然后利用Rank模型对候选广告进行排序。这些广告要么显式要么隐式的和用户query相关联。但是在Facebook这样的社交场合中，广告并没有和用户query相关联，但是用户看到的广告一定程度上反映了用户的人口统计特性和兴趣特性。基于这个原因，在Facebook上展示的广告相比于搜索广告中的要多一些。

在实际的生产环境中，为每个用户确定广告候选是一件系统性设施工作，Facebook主要通过做多个分类器级联来实现。但是论文中分析的是最后的那一个prediction模型。它直接给出最后的CTR概率。

1.2 研究背景

CTR预估（Click-Through Rate Prediction）是互联网计算广告中的关键环节，预估准确性直接影响公司广告收入。CTR预估中用的最多的模型是LR（Logistic Regression），LR是广义线性模型，与传统线性模型相比，LR使用了Logit变换将函数值映射到0~1区间，映射后的函数值就是CTR的预估值。LR这种线性模型很容易并行化，处理上亿条训练样本不是问题，但线性模型学习能力有限，需要大量特征工程预先分析出有效的特征、特征组合，从而去间接增强LR的非线性学习能力。

LR模型中的特征组合很关键， 但又无法直接通过特征笛卡尔积解决，只能依靠人工经验，耗时耗力同时并不一定会带来效果提升。如何自动发现有效的特征、特征组合，弥补人工经验不足，缩短LR特征实验周期，是亟需解决的问题。Facebook 2014年的文章介绍了通过GBDT（Gradient Boost Decision Tree）解决LR的特征组合问题，随后Kaggle竞赛也有实践此思路，GBDT与LR融合开始引起了业界关注。 GBDT（Gradient Boost Decision Tree）是一

种常用的非线性模型，它基于集成学习中的boosting思想，每次迭代都在减少残差的梯度方向新建立一颗决策树，迭代多少次就会生成多少颗决策树。GBDT的思想使其具有天然优势可以发现多种有区分性的特征以及特征组合，决策树的路径可以直接作为LR输入特征使用，省去了人工寻找特征、特征组合的步骤。

这种通过GBDT生成LR特征的方式（GBDT+LR），业界已有实践（Facebook，Kaggle-2014），且效果不错，是非常值得尝试的思路。

2. 评估函数

论文目的是分析机器学习模型的影响因素，所以没有使用实际利益相关的评测函数。而是主要从以下两方面进行：

- Normalized Cross-Entropy 或者叫做 Normalized Entropy， 缩写NE
- Calibration 校准

2.1 Normalized Cross-Entropy (NE)

NE的公式如下：

$$NE = \frac{-\frac{1}{N}\sum_{i=1}^{n}(\frac{1+y_i}{2}log(p_i) + \frac{1-y_i}{2}log(1-p_i))}{-(p*log(p)+(1-p)*log(1-p))}$$

参数解释： N: 样本数； y_i: 训练集标签； p: 平均历史点击率；

- NE等于预测的log loss除以background CTR的熵
- NE越小模型性能越好
- 除去background CTR的熵，使得NE对background CTR不敏感
- p代表平均经验CTR

2.2 Calibration

- Calibration校准是平均预测CTR与经验CTR的比值，它是一个比例。
- Calibration越接近1，模型性能越好

AUC也是一个非常不错的评价指标，但是它有个问题。比如当我们的模型预测的CTR概率都偏高了2倍，我们可以通过Calibration校准，使用一个全局的0.5的系数来修正。修正之后NE也会提高，而AUC却保持不变。 在实际工作中，我们希望得到的是尽可能准确的预测每个广告被点击的概率，而不是仅仅得到相对的概率排序。所以AUC不如上面的NE、Calibration合适。

3. 模型架构

经过多次实验，FB得出结论：**正确的模型 + 强特征**是提升模型性能的核心。相比于这两点，其他的因素的影响就小很多，比如学习速率、采样率等。当数据量足够大时，一个好的模型应该是稳定的，也就说参数的调整不会导致模型性能出现剧烈的震荡。

这里面，正确的模型就是指：**Logistic Regression + Boosting Decision Tree**。特征的话包含两方面的特征：**用户或广告的历史信息特征、上下文特征**。其中，**用户或广告的历史信息特征取决定性作用**。

学习算法是用的是**Stochastic Gradient Descent(SGD)**，或者 **Bayesian online learning scheme for probit regression(BOPR)** 都可以。但是最终选择的是SGD，原因是资源消耗要小一些。 SGD和BOPR都可以针对单个样本进行训练，所以他们可以做成流式的学习器(stream learner)。

3.1 决策树Feature Transforms

为了提升线性分类器的准确度，有两种方法进行特征变换：

1. 对于连续特征。先进行离散化bin，然后把bin的index作为离散型特征。这样的话，线性模型可以分段的学习到一个非线性的映射，在每一段内的映射是不变的。另外，对于bin边界的学习非常重要；
2. 对于离散特征。做笛卡尔积，生成的是**tuple input features**。笛卡尔积穷举了所有的特征组合，其中也包含部分没用的组合特征，不过可以筛选出来（只有那些不能被修剪掉的特征组合才是有用的）；其次，如果输入特征是连续的，也可以使用联合分箱，如使用k-d tree。

> 笛卡尔乘积是指在数学中，两个集合X和Y的笛卡尔积（Cartesian product），又称直积表示为$X \times Y$，第一个对象是 X 的成员而第二个对象是 Y 的所有可能有序对的其中一个成员。 假设集合A={a, b}，集合B={0, 1, 2}，则两个集合的笛卡尔积为{(a, 0), (a, 1), (a, 2), (b, 0), (b, 1), (b, 2)}。
>
> 笛卡儿积得名于笛卡儿，因为这概念是由他建立的解析几何引申出

提升决策树(boosted decision tree)就可以很方便很好的实现上面我们说的这种非线性和tuple特征变换。**对于一个样本，针对每一颗树得到一个类别型特征。该特征取值为样本在树中落入的叶节点的编号。**举例来说：

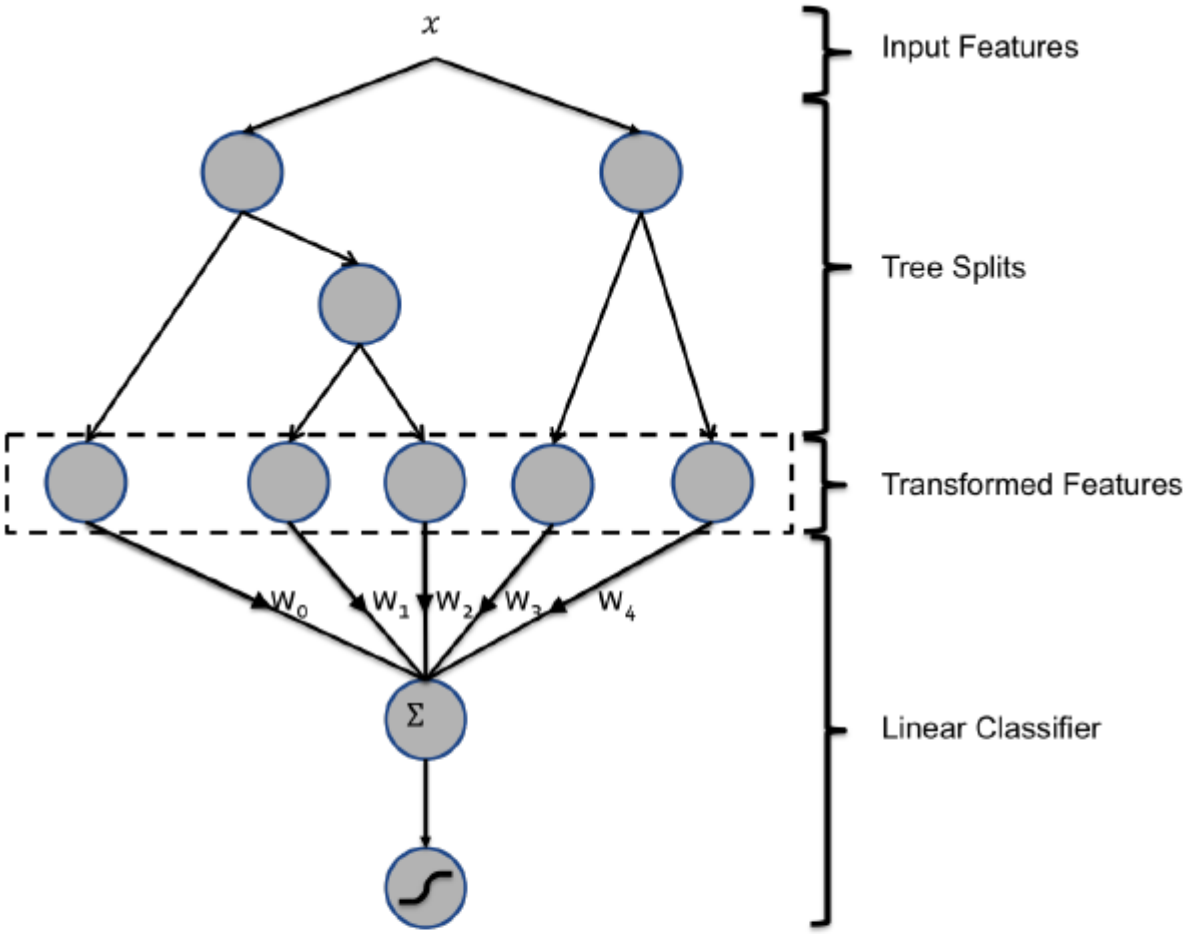


Figure 1: Hybrid model structure. Input features are transformed by means of boosted decision trees. The output of each individual tree is treated as a categorical input feature to a sparse linear classifier. Boosted decision trees prove to be very powerful feature transforms.

上图中的提升决策树包含两棵子树，第一棵树包含3个叶节点，第二棵树包含2个叶节点。输入样本x，在两棵树种分别落入叶子节点2和叶子节点1。那么特征转换就得到特征向量 `[0 1 0 1 0]`。也就是说，把叶节点编号进行one-hot编码。

那么， 怎么样直观的理解这种特征变化：

1. 看做是一种有监督的特征编码。把实值的vector转换成紧凑的二值的vector。
2. 从根节点到叶节点的一条路径，表示的是在特征上的一个特定的规则。所以，叶节点的编号代表了这种规则。表征了样本中的信息，而且进行了非线性的组合变换。
3. 最后再对叶节点编号组合，相当于学习这些规则的权重。

从最后的实验结果来看：将LR和GBDT进行组合模型的性能指标（NE）相比于没有经过数进行特征转化的标准交叉熵（NE）提升超过了3.4%（相比于特征工程只能在千分位上对NE进行降低，可以看出，GBDT+LR这种组合具有一个非常显著的提升）！

Table 1: Logistic Regression (LR) and boosted decision trees (Trees) make a powerful combination. We evaluate them by their Normalized Entropy (NE) relative to that of the Trees only model.

Model Structure	NE (relative to Trees only)
LR + Trees	96.58%
LR only	99.43%
Trees only	100% (reference)

> GBDT模型的特点，非常适合用来挖掘有效的特征、特征组合。业界不仅GBDT+LR融合有实践，GBDT+FM也有实践，2014 Kaggle CTR竞赛冠军就是使用GBDT+FM，可见，使用GBDT融合其它模型是非常值得尝试的思路。

3.2 Data freshness

论文里的数据取2013年某一周内的实际数据，并且尽可能的保证线上线下的数据分布是一致的。训练集、测试集的划分基本都是按照时间来的，比如选一天的数据作为训练集，其后的一天或者几天作为测试数据。

CTR系统的环境经常变化，**数据的分布**也经常随着时间变化而变化。为了验证 **data freshness** 对模型的影响，实验中训练集固定为某一天的数据，然后分别测试在之后连续六天的模型的表现。

Figure 2: Prediction accuracy as a function of the delay between training and test set in days. Accuracy is expressed as Normalized Entropy relative to the worst result, obtained for the trees-only model with a delay of 6 days.

可以发现随着天数的增加，**data freshness** 也变得越来越差，模型的性能也越来越差。所以，针对每天的偏差进行重新训练就非常有必要。

一种做法是说每天都重新训练。即使是mini-batch来训练，也会非常耗时。提升树的训练时间受很多因素的影响，比如：样本数量、树深度、树数量、叶子节点个数等。为了加快速度，可以在多CPU上通过并行化来实现。

那么现在我们给出一种新的方法，可以做到：

1. 提升树可以一天或者几天来训练一次
2. LR可以实现在线学习*online learning*，几乎是实现实时的训练

3.3 LR线性分类器

为了最大化data freshness，我们采取的措施是针对Logistic Regression进行在线增量训练。也就是说只要用户点击了广告，生成了新的样本，就进行增量训练。

为此，Facebook针对*SGD-based online learning*研究了5中学习速率的设置方式，如下：

1. Per-coordinate learning rate: The learning rate for feature i at iteration t is set to:

$$\eta_{t,i} = \frac{\alpha}{\beta + \sqrt{\sum_{j=1}^{t} \nabla_{j,i}^2}}$$

   α, β 是两个可调参数；

2. Per-weight square root learning rate:

$$\eta_{t,i} = \frac{\alpha}{\sqrt{n_{t,i}}}$$

   其中 $n_{t,i}$ 特征为 i 直到迭代轮 t 的总的训练实例；

3. Per-weight learning rate:

$$\eta_{t,i} = \frac{\alpha}{n_{t,i}}$$

4. Global learning rate:

$$\eta_{t,i} = \frac{\alpha}{\sqrt{t}}$$

5. Constant learning rate:

$$\eta_{t,i} = \alpha$$

1. 前三种使得不同的参数有不同的学习速率
2. 后两种对于所有的参数都是用相同的学习速率

最终的实验结果是：**Per-coordinate learning rate效果最好：**

这个跟Adagrad的做法几乎一样，分母上使用梯度的平方进行累加，然后开根号。使得不同的参数具有不同的学习速率。

> 顺便提一句，Adagrad也有缺点：随着迭代不断进行，学习速率无限的减小，直到模型无法进行学习。

实验参数设置及对应结果：

Table 2: Learning rate parameter

Learning rate schema	Parameters
Per-coordinate	$\alpha = 0.1, \beta = 1.0$
Per-weight square root	$\alpha = 0.01$
Per-weight	$\alpha = 0.01$
Global	$\alpha = 0.01$
Constant	$\alpha = 0.0005$

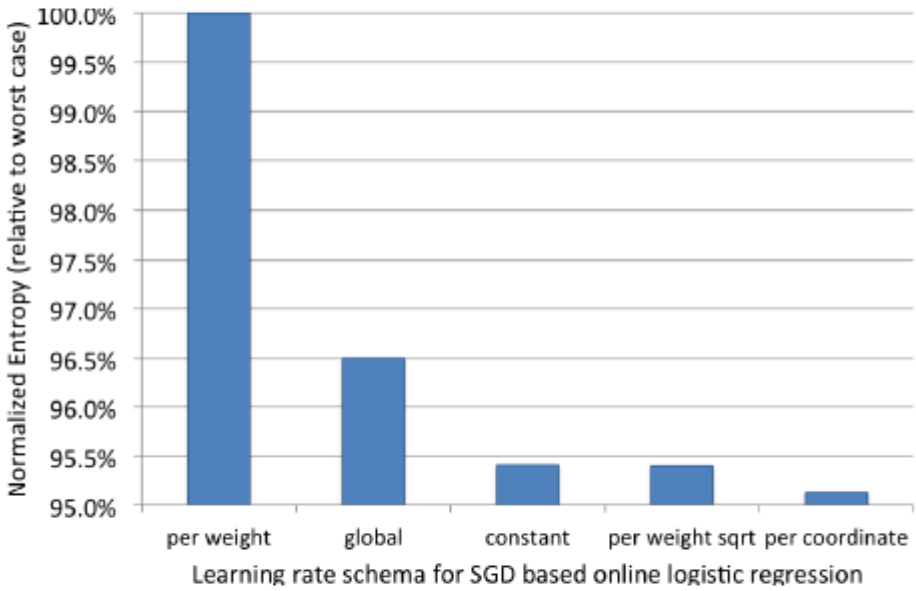


Figure 3: Experiment result for different learning rate schmeas for LR with SGD. The X-axis corresponds to different learning rate scheme. We draw calibration on the left-hand side primary y-axis, while the normalized entropy is shown with the right-hand side secondary y-axis.

其中，global learning rate失败主要是由于训练样本在特征上分布不平衡造成。

> 因为每个训练实例包含不同特征，而一些比较popular的特征相比于其他特征可以对应更多的训练实例，在global learning rate策略下，会造成对只有很少实例在该特征上为非零值的特征学习率下降太快，阻止了该特征收敛到最佳权重。

而对于per-weight learning rate策略本身是用于解决驯良样本在特征上分布不平衡问题的，这里依旧失败的原因是对于所有特征，学习率下降太快，训练在模型收敛到次优点的情况下过早终止。

另外，之前提到的BOPR和使用per-coordinate的SGD的表现是非常相似的。他们的效果也非常接近，但是BOPR需要计算均值和方差，计算量更大。两者效果比较如下：

Table 3: Per-coordinate online LR versus BOPR

Model Type	NE (relative to LR)
LR	100% (reference)
BOPR	99.82%

> SGD + pre-coordinate learning rate 和BOPR的效果差不多，但胜在策略简单，时间复杂度空间复杂度都完胜BOPR。

LR 相比 BOPR的优势：

考虑在LR中，每个稀疏要素值只有一个权重, 而不是一个平均值和方差，LR model size只有BOPR的一半。

BOPR 相比于 LR的优势：

它提供了一个完整的预测分布的概率点击。这可用于计算预测分布的百分位数, 可用于探索学习方案 [3]。

4. 线上模型架构

这部分主要是说明 **online data joiner**。前面我们研究过 **data freshness** 对于模型的训练是非常重要的。那么新的训练数据是怎么产生的呢？这就是 **online data joiner** 的作用。

这里最关键的步骤就是把 labels(click/no-click) 和训练输入 (ad impressions) 以一种在线的方式连起 (join) 起来。所以系统被称为 **online data joiner**。

4.1 label标注

首先设定一个足够长的阈值。一个广告展示给用户之后，如果用户在阈值的时间内没有点击广告就标记为 **no-click**，点击了的话就标记为 **click**。这个等待的时间窗口需要非常小心的调整。 如果太长了，会增加缓存 impression 的内存消耗，而且影响实时数据的产生；如果太短了则会导致丢失一部分的点击样本，会影响 **click converage (点击覆盖)**。

click converage (点击覆盖) 表示有多少个点击行为被记录了下来生成了样本。**online data joiner** 必须保证尽可能高的点击覆盖，也就是尽可能多的来记录下来所有的点击行为。但是如果等待太久就会增加缓存开销等影响。所以 **online data joiner** 必须在 **click converage** 和资源消耗之间做出平衡，又一个**trade-off**。

如果点击覆盖比较低，意味着很多用户的点击不但没有记录下来，而是变成了没有点击。造成数据分布发生偏差，结果就是：**模型学习到的CTR值要比真实值低很多**。不过实际情况中，问题比较好解决：增大等待时间窗口，只要内存消耗还可以接受就行。

4.2 模型架构

Online data joiner 系统结构如下：

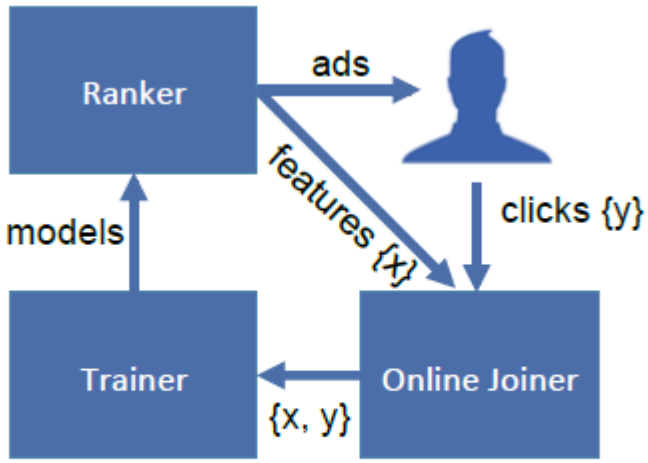


Figure 4: Online Learning Data/Model Flows.

广告展示生成特征，用户给出反馈：点击或者未点击。Online Joiner捕获反馈生成新的训练样本，训练样本经过Trainer的学习得到新的模型。模型反过来影响Ranker系统对展示的广告进行选择排序，用户又看到了新的广告，决定是否要点击。一直这样下去，形成一个**闭环系统**。

4.3 挑战

系统异常是在线学习系统的一大挑战。这里的异常就是指系统异常，比如系统出现问题导致stream data是老数据。可能分类器就会学习到错的数据，针对所有的点击率都给出一个非常低甚至是0的概率。这显然不是我们想看到的。可以依靠一些 **保护机制**来解决，比如：当发现实时的训练数据分布发生比较大变化的时候，就把 online trainer 和 online joiner 自动断开，防止Trainer学习到坏的数据分布。

5. 内存占用和延迟

所有的这些探索都是为了能够平衡模型性能(accuracy)和资源消耗(内存、CPU)。只有当你充分了解模型和数据每个部分后，才能根据实际情况做出最佳的取舍。

5.1 Number of boosting trees

下图给出了，boosting trees对模型的影响：

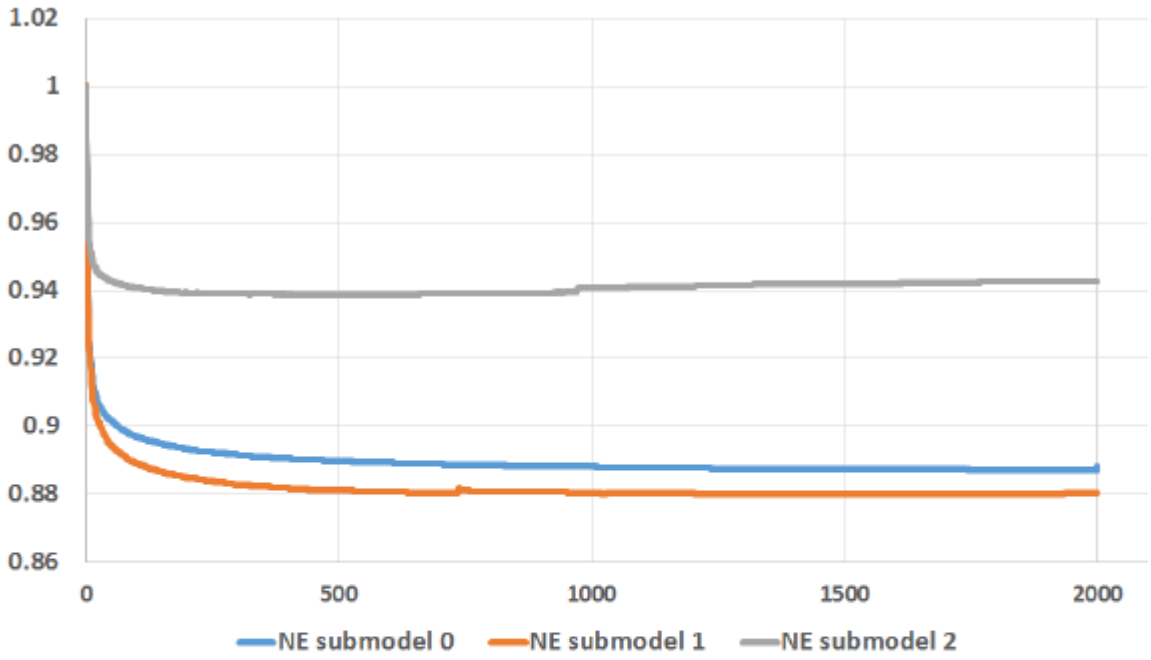


Figure 5: Experiment result for number of boosting trees. Different series corresponds to different sub-models. The x-axis is the number of boosting trees. Y-axis is normalized entropy.

boosting tree数量从1到2000，叶节点个数被限制为最大12个。submodel之间的区别在于训练数据大小的不同，比如submodel 2的训练数据只有前面两个的1/4。 可以看到随着boosting tree数量的增加，模型的性能有所提升。但是几乎所有的提升都来自于前500个trees，而后面的1000个trees的提升甚至都不到0.1%。submodel 2在1000颗trees甚至模型效果在变差，原因是出现过拟合。

5.2 Boosting feature importance

为了在资源消耗和模型性能之间做到平衡，可以通过控制Feature Count来调节。如果想删掉一些特征的话，那么就需要研究这些特征的重要程度的分布，并研究删除部分特征后的效果。

下图给出了特征重要程度的分布情况：

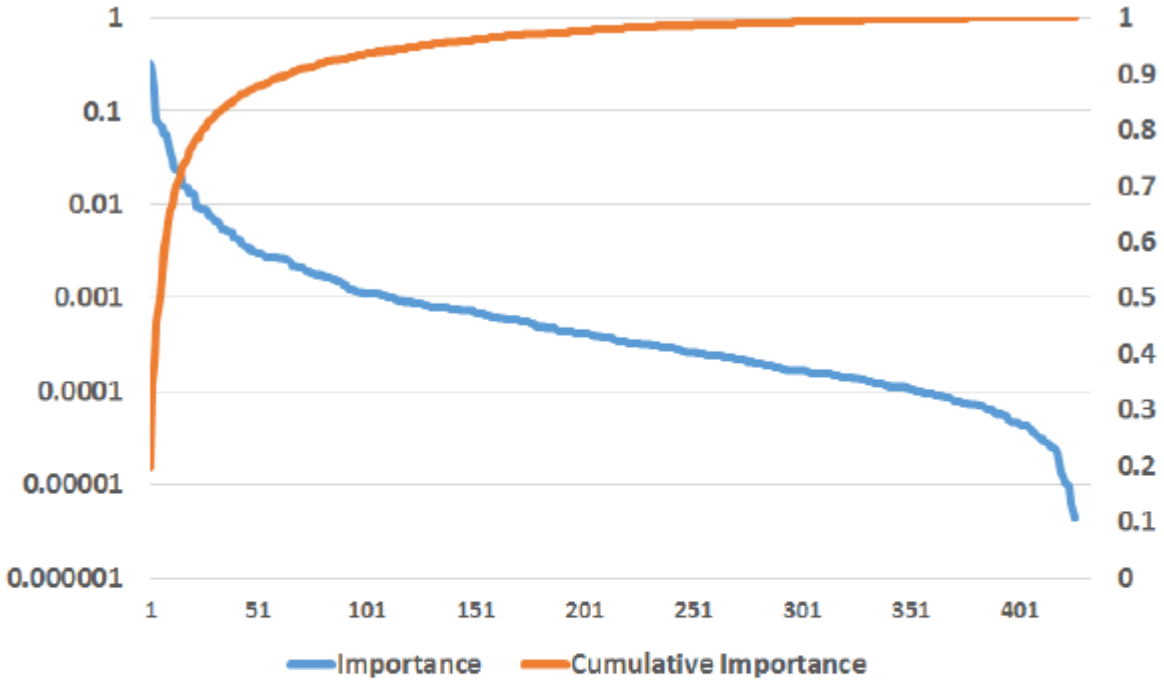


Figure 6: Boosting feature importance. X-axis corresponds to number of features. We draw feature importance in log scale on the left-hand side primary y-axis, while the cumulative feature importance is shown with the right-hand side secondary y-axis.

上图首先对特征按照重要程度来进行排序，编号后再画图。特征重要程度按照**使用该特征进行分裂，所带来的loss减小的累积量**。因为一个特征可以在多颗树上进行使用，所以累积要在所有的树上进行。

上图中，黄线表示对特征进行累加后的值，然后进行log变换。可以看到最终结果是1，表示所有特征的重要度总和是1. 最重要的是期初非常陡峭，上升的非常快，说明**特征重要度主要集中在top10这些特征中**。前10个特征，贡献了50%的重要度，后面300个特征，贡献了1%的重要度。

作者又做实验验证了只保留前10、20、50、100以及200特征时，模型表现如下：

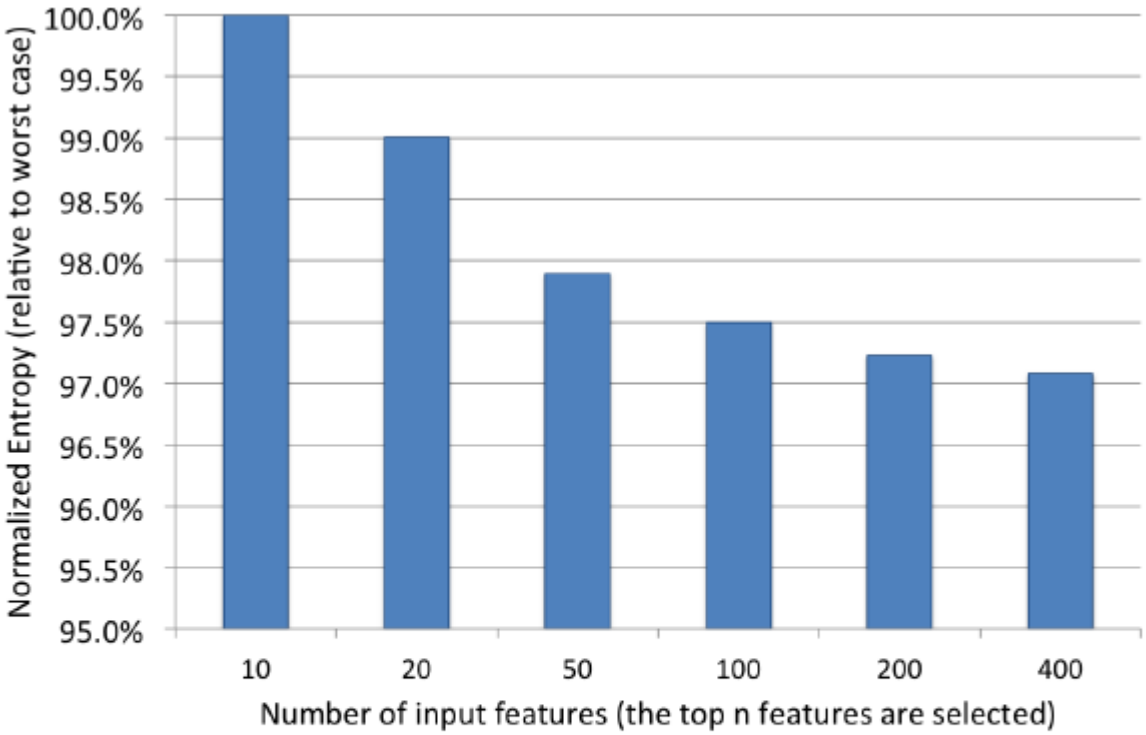


Figure 7: Results for Boosting model with top features. We draw calibration on the left-hand side primary y-axis, while the normalized entropy is shown with the right-hand side secondary y-axis.

当我们包含了更多的特征时候, 标准交叉熵（NE）具有相似的递减回归属性（即随着特征的增大，NE降低速率逐步降低至0）。

5.3 Historical features VS Context features

针对两大类特征：历史信息特征（用户+广告）、上下文特征。论文还研究了这两类特征对模型性能的贡献程度。先给出结论：**历史信息特征占主导地位**。

实验结果如下：

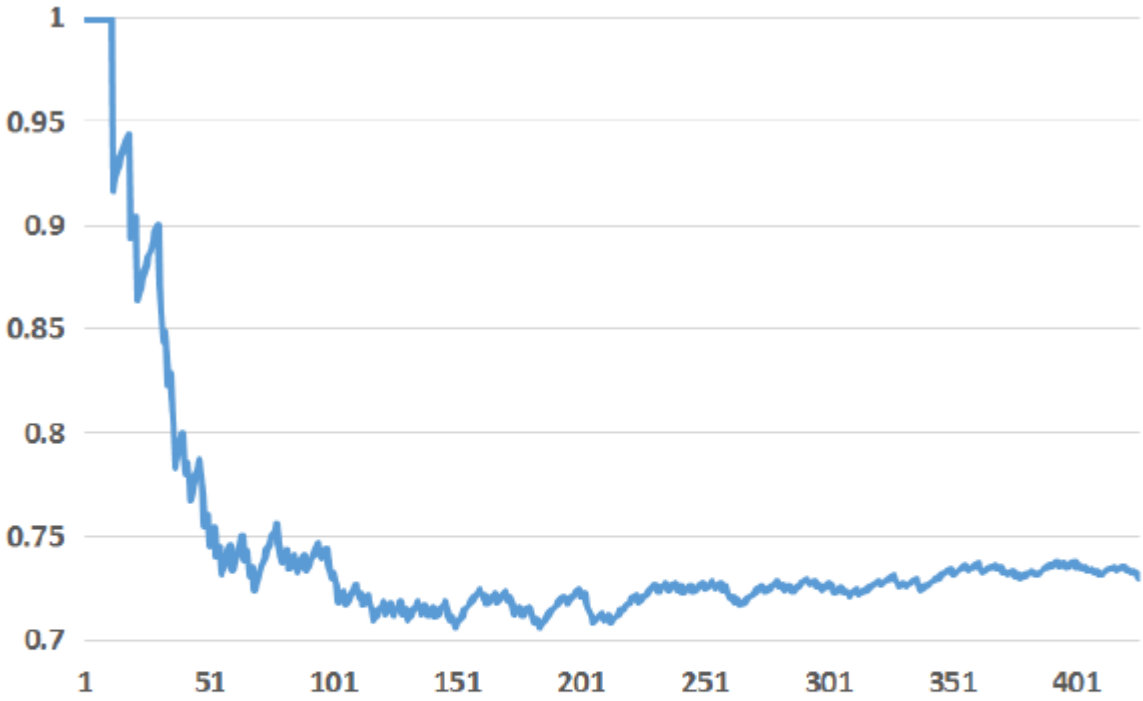


Figure 8: Results for historical feature percentage. X-axis corresponds to number of features. Y-axis give the percentage of historical features in top k-important features.

同样，先把特征按照重要程度排序，再画图。横轴是特征数量，纵轴是 historical 特征在 top k 个重要特征中所占的百分比。可以看到前10个特征中，全是历史信息特征；前20个特征中，只有2个上下文特征。所以：**历史信息特征比上下文特征重要太多了**。

- 历史信息特征。主要是指用户或者广告之前的一些信息，比如：该广告上周的CTR值、该用户的历史平均CTR值等
- 上下文特征。比如：用户使用的设备、当前页面、时间、一周第几天等

由于Facebook的数据非常敏感，论文里不能提供具体的特征都有哪些。

论文中还研究了单独使用这两类特征的效果：

Table 4: Boosting model with different types of features

Type of features	NE (relative to Contextual)
All	95.65%
Historical	96.32%
Contextual	100% (reference)

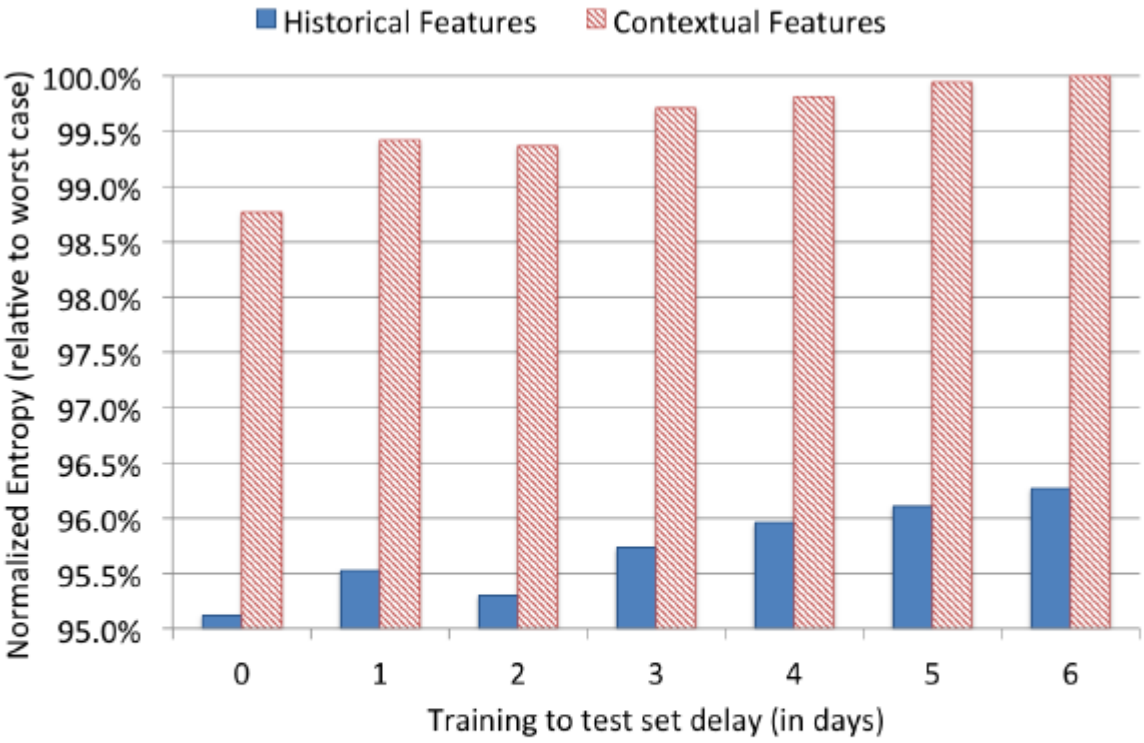


Figure 9: Results for datafreshness for different type of features. X-axis is the evaluation date while y-axis is the normalized entropy.

和之前的结论保持一致。而且还可以发现使用Historical特征的模型，对data freshness的依赖相对要小一些。这也和我们的直观理解是相符的：历史信息特征包含用户长时间的行为特征，相比于上下文特征更加稳定。

但是，**上下文特征在解决冷启动问题上有优势**。对于新的用户或者广告，上下文特征对于给出一个合理的CTR预测是必不可少的。

6. 处理大量训练数据

很多的计算广告领域的训练数据量都是非常巨大的，那么如何有效的控制训练带来的开销就非常重要。常用的办法是**采样**，分为：

- Uniform Subsampling
- Negative down sampling

6.1 Uniform subsampling

均匀采样非常的简单，易于实现。而且使用均匀采样没有改变训练数据的分布，所以模型不需要修改就可以直接应用于测试数据上。 下图给出了不同采样率对模型性能的影响：

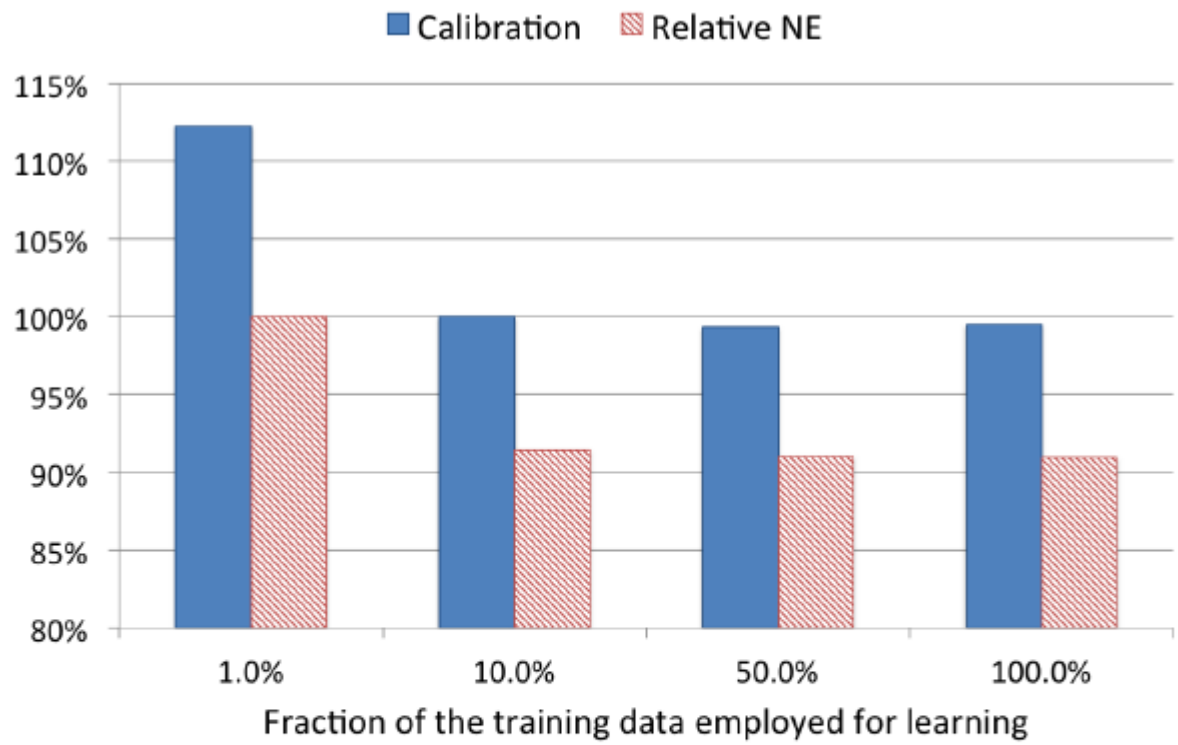


Figure 10: Experiment result for data volume. The X-axis corresponds to number of training instances. We draw calibration on the left-hand side primary y-axis, while the normalized entropy is shown with the right-hand side secondary y-axis.

可以看到更高的采样率使用了更多的训练数据，提升了模型的效果。从图中可以看到使用10%的数据，相比于使用100%的数据，仅仅造成了1%的性能降低。是非常小的。 对于Calibration校准，均匀采样不会造成影响。

6.2 Negative down sampling

计算广告中大部分的训练样本都极度不平衡，这对模型会造成很大影响。一种解决办法就是对负样本进行欠采样。实验结果如下：

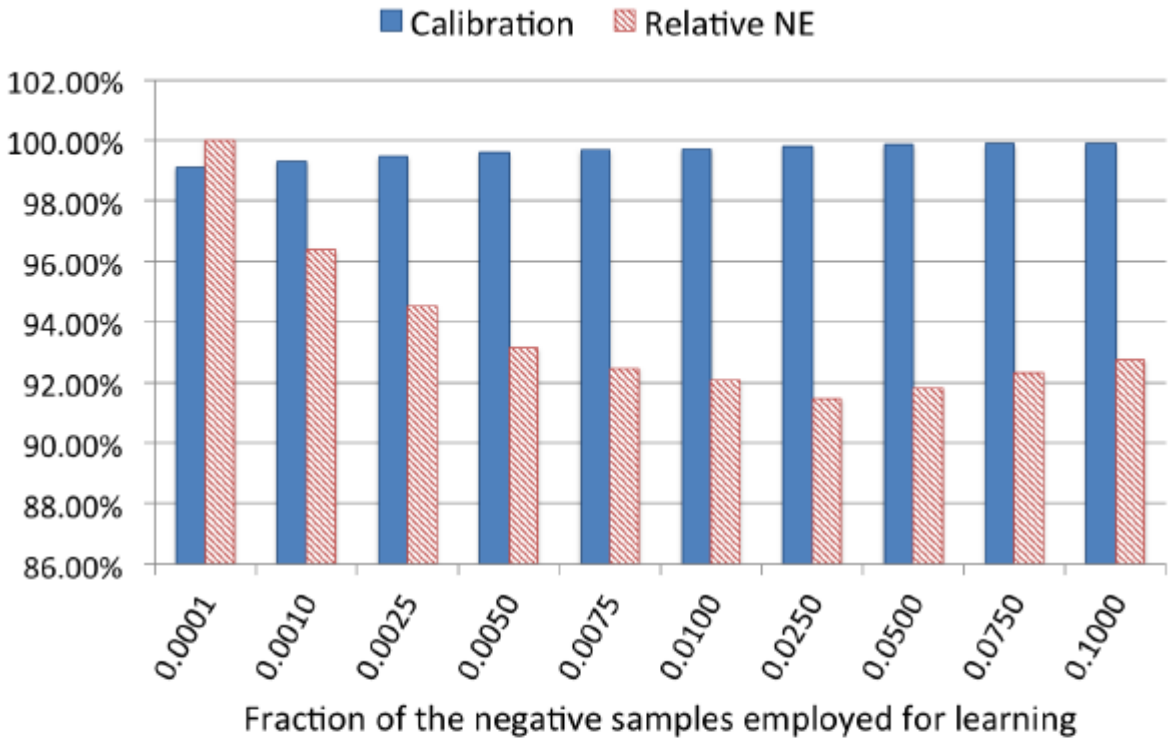


Figure 11: Experiment result for negative down sampling. The X-axis corresponds to different negative down sampling rate. We draw calibration on the left-hand side primary y-axis, while the normalized entropy is shown with the right-hand side secondary y-axis.

可以看到采样率不同，对模型性能影响很大。采样率为0.025的时候取得最好结果。

6.3 Model Re-Calibration

负样本欠采样可以加快训练速度并提升模型性能。但是同样带来了问题：**改变了训练数据分布**。所以需要进行校准。 举例来说，采样之前CTR均值为0.1%，使用0.01采样之后，CTR均值近似变为10%。我们需要对模型进行Calibration(校准)使得模型在实际预测的时候恢复成0.1%。调整公式如下：

$$q = \frac{p}{p + (1-p)/w}$$

其中：

- w是采样率
- p是在采样后空间中给出的CTR预估值
- 计算得到的q就是修正后的结果

7. 总结

Facebook提出的LR + GBDT来提取非线性特征进行特征组合的方式非常经典，主要特性总结如下：

- Data Freshness很重要。模型至少一天需要重新训练一次；
- 使用Boosted Decision Tree进行特征转换很大程度上提高了模型的性能；
- 最好的在线学习方法：LR + per-coordinate learning rate；

关于平衡计算开销和模型性能所采用的技巧：

- 调整Boosted decision trees数量；
- 去掉部分重要性低的特征，对模型的影响比较小；
- 相比于上下文特征，用户/广告历史特征要重要的多；
- 针对大量训练数据可以进行欠采样。

8. Note

8.1 GBDT建树细节

建树采用GBDT而非RF（Random Forests）。解读如下：

1） 为什么建树采用ensemble决策树？

一棵树的表达能力很弱，不足以表达多个有区分性的特征组合，多棵树的表达能力更强一些。GBDT每棵树都在学习前面棵树尚存的不足，迭代多少次就会生成多少颗树。按paper以及Kaggle竞赛中的GBDT+LR融合方式，多棵树正好满足LR每条训练样本可以通过GBDT映射成多个特征的需求。

2） 为什么建树采用GBDT而非RF？

RF也是多棵树，但从效果上有实践证明不如GBDT。且GBDT前面的树，特征分裂主要体现对多数样本有区分度的特征；后面的树，主要体现的是经过前N颗树，残差仍然较大的少数样本。优先选用在整体上有区分度的特征，再选用针对少数样本有区分度的特征，思路更加合理，这应该也是用GBDT的原因。

然而，Facebook和Kaggle竞赛的思路是否能直接满足现在CTR预估场景呢？

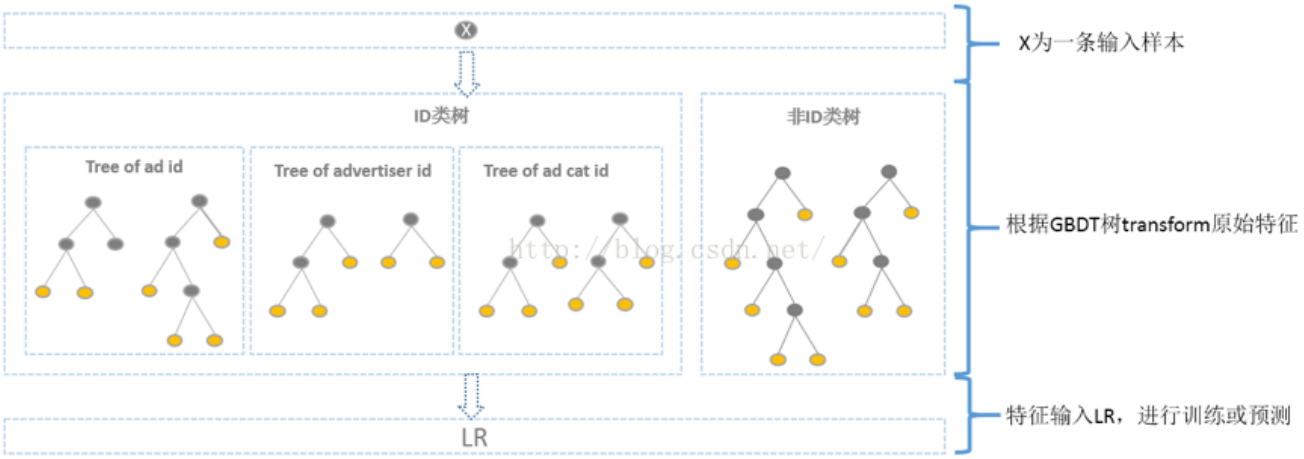
按照Facebook、Kaggle竞赛的思路，不加入广告侧的Ad ID特征？但是现CTR预估中，Ad ID类特征是很重要的特征，故建树时需要考虑Ad ID。直接将Ad ID加入到建树的feature中？但是Ad ID过多，直接将Ad ID作为feature进行建树不可行。下面将介绍针对现有CTR预估场景GBDT+LR的融合方案。

8.2 GBDT与LR融合方案

AD ID类特征在CTR预估中是非常重要的特征，直接将AD ID作为feature进行建树不可行，顾考虑为每个AD ID建GBDT树。但互联网时代长尾数据现象非常显著，广告也存在长尾现象，为了提升广告整体投放效果，不得不考虑长尾广告[12]。在GBDT建树方案中，对于曝光充分训练样本充足的广告，可以单独建树，发掘对单个广告有区分度的特征，但对于曝光不充分样本不充足的长尾广告，无法单独建树，需要一种方案来解决长尾广告的问题。

综合考虑方案如下，使用GBDT建两类树，非ID建一类树，ID建一类树。1）非ID类树：不以细粒度的ID建树，此类树作为base，即便曝光少的广告、广告主，仍可以通过此类树得到有区分性的特征、特征组合。2）ID类树：以细粒度的ID建一类树，用于发现曝光充分的ID对应有区分性的特征、特征组合。

如何根据GBDT建的两类树，对原始特征进行映射？以如下图3为例，当一条样本x进来之后，遍历两类树到叶子节点，得到的特征作为LR的输入。当AD曝光不充分不足以训练树时，其它树恰好作为补充。



8.3 如何使用GBDT 映射得到的特征？

通过GBDT生成的特征，可直接作为LR的特征使用，省去人工处理分析特征的环节，LR的输入特征完全依赖于通过GBDT得到的特征。此思路已尝试，通过实验发现GBDT+LR在曝光充分的广告上确实有效果，但整体效果需要权衡优化各类树的使用。同时，也可考虑将GBDT生成特征与LR原有特征结合起来使用，待尝试。

参考目录

[1] Practical lessons from predicting clicks on ads at facebook

[2] Study: Practical Lessons from Predicting Clicks on Ads at Facebook

[3] Facebook经典模型LR+GBDT理论与实践

[4][CTR预估中GBDT与LR融合方案](https://blog.csdn.net/lilyth_lilyth/article/details/48032119)